VOL. 11, NO. 18, SEPTEMBER 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

TWO WHEEL SELF-BALANCING VEHICLE USING ARDUINO

Muhammad Ikram Mohd Rashid, Law Choon Chuan and Suliana Ab Ghani
Faculty of Electrical Engineering, Universiti Malaysia Pahang, Pekan, Pahang, Malaysia
E-Mail: mikram@ump.edu.my

ABSTRACT

This paper concerns about the implementation of two wheel self-balancing vehicle using Arduino. Tilt angle and motor speed rate are functioning as input of the system to perform balancing of the vehicle. Inertia Measurement Unit (I.M.U.) and DC motors were used as sensor and actuator respectively for this system. Moreover, the control of vehicle system used PID controller and implemented in Arduino board. This project is represents and focuses on power drive system because it involves a series of power drive and embedded controllers. The hardware of the vehicle is being produced and tested in laboratory.

Keywords: inertia measurement unit, PID controller, arduino.

INTRODUCTION

Self-balancing two wheel electric vehicle refers to a type of personal transport that performs balanced transportation with only two wheels. Typically, balancing vehicle refers to vehicle that can balance by itself without assistance of external forces. Though, the statement mentioned only balancing but not transportation for the vehicle. As technology era is improving, two wheel balancing vehicle is now being developed into various types of self-transportation that not only balance by two wheels but also support for transportation as well. The vehicle is developed using inverted pendulum concept where covers up for both movement and stabilization [1].

Two wheel self-balancing electric vehicle using Arduino, is a project that studies the characteristic of two wheel balancing vehicle while construct an algorithm that links between microcontroller, balancing sensor and acceleration sensor to perform a self-balancing two wheel electric vehicle that covers functions for balance transportation.

DESIGN BACKGROUND

There are many types of two wheel balancing robot being introduced in the market, modeling are mostly various depend on personal criteria on personal transporter. Modeling mostly refers to Segway liked personal transport that was the first and based design for first prototype of two wheel balancing vehicle. Typical two wheel balancing vehicle is made up of base with two parallel positioned wheels on both left and right side while a steer functioned rod is positioned at forward front part of the base, enabling directional controller for end user to determine rotation angle.

Figure-1 illustrates the modeling of two wheel balancing vehicle where the structure clearly indicates that initial position of the modeling is unstable. Positioning of two round shaped wheel at the base of the vehicle was purposed to enable motion to structure so that tuning can be made when the vehicle is out of balance. By referring to [1], [2] and [3] case study, it showed that modeling generations are mostly refers to Lagrange equation regardless of axis.

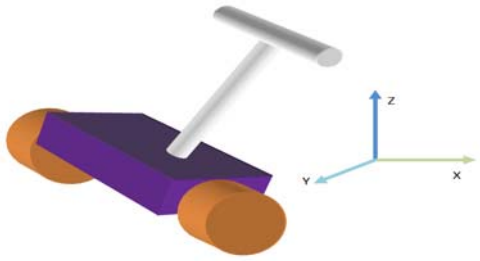

Figure-1. Modeling of two wheel balancing vehicle.

By referring to Figure-1, it can be explained that trajectory of balancing only refers to the tilting of x-z axis and the anti-trajectory concept to balance the vehicle back. It can be simplified that when the vehicle is tilting forward, an oppose force to counter tilt angle is needed so that vehicle can be maintain in balance condition.

As shown in Figure-2, it displayed the movement of vehicle when vehicle traces tilt angles from sensor. When the vehicle is tilt forward, vehicle will move forward with a tuning speed in order to keep balance tilt forward. The same condition goes to tilt backward condition with a balance tilt backward motor rotation. During stationary condition, there will not be having movement but just static balancing on the spot until tilt angle is traced. These trajectories acceleration will remain as the sensor is sending signal to microcontroller and motor is react to changes of sensor values leading to a smooth graceful balancing acceleration with a proper feedback controller. As mentioned in [6], trajectories continuous movements with boundaries are consider as graceful where it leads to smooth actuation.

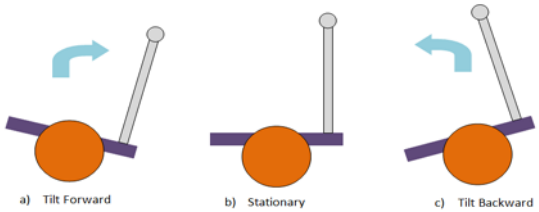

Figure-2. Trajectory of vehicle movement against tilt angle.

SIMULATION RESULTS

The prototype is simulated based on the wiring connection as depicted in Figure-3 and the functions of system block is illustrated in Figure-4.

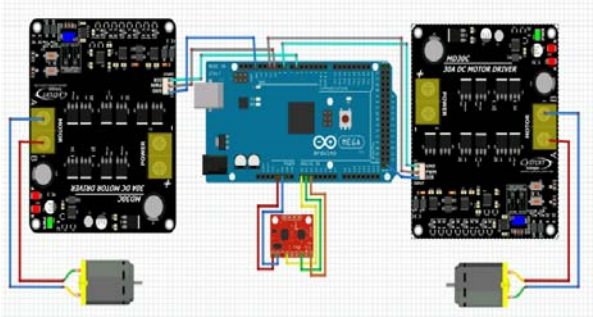

Figure-3. Complete circuit connection.

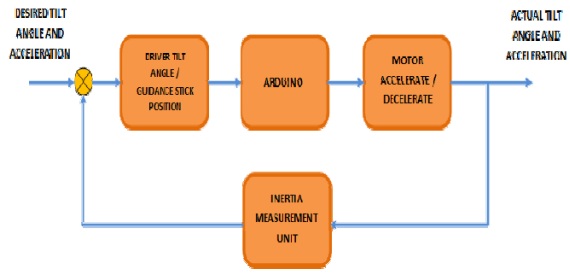

Figure-4. Overall system block diagram.

Raw program results are being tested before the complete connection is made. These results indicates the values of readings calibrated when sensor is twisted in 0°, 90°, 180° and -90°. From Figure-5, it shows that sensor is being placed at 0° with acceleration in x-axis for 1 to 2, while acceleration in z-axis is in 103 where positive gravity force facing down. For gyro y, since there is no rotation beyond the axis, so the value will be around 0 to 1.

Figure-5. Axis acceleration for 0°.

When IMU is tilted at 90°, the result is changing as in Figure-6. Acceleration in x-axis changed to range in between 103 to 105. Meanwhile, acceleration in z-axis have deducted into half due to changes in gravitational force. The value stated in this state is around 0 to 3. Gyro in y-axis remained the same due to no actual rotation for the axis.

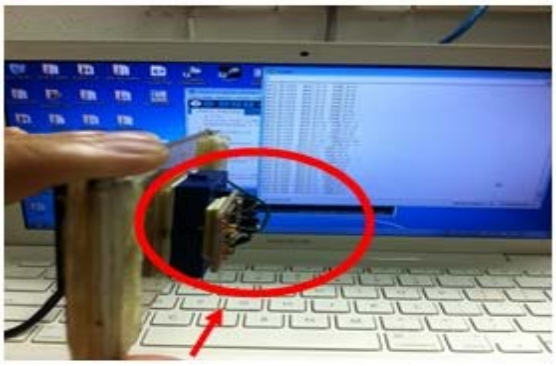

Figure-6. Axis acceleration for 90°.

Once again the tilt angle is twisted, and this time the value changes again. From Figure-7, it shows that acceleration in x-axis increased to range in between -4 to -8. Due to reversal gravitational force, acceleration in z-axis changed into negative values in the range between -95 to -97. In this tilt angle, y-axis remained unchanged.

Figure-7. Axis acceleration for 180°.

For the last turn, the IMU now state in -90°, where acceleration values in x-axis drops down somewhere around range within -99 to -100. For this condition z-axis acceleration changed back to positive values indicating the values of 13 to 15. As well as others, gyro in axis-y remained unchanged. The data collected is well described in Figure-8.

Figure-8. Axis acceleration for -90°.

Upon hardware completion, program is tuned with real actual hardware testing. Since angle acquisition is completed within program, the next important part is the tuning of PID for motor response. As stated earlier, the project lacks of parameter in generating linear functions for motor control equations. Hence, the tuning method has to be done by using manual method. For the first test, gain for all parameters is set to 0 while overall control gain is set to 1. A free drop test is performed and the result is being plotted in Figure-9. From the response, it is clearly seen that motor response is not fast enough to change of angle. From results, it showed that motor speed start to response when angle change reaches 3 degree at 11 milliseconds. The data indicates that motor is delayed 11 milliseconds to react for angle changes which are consider slow response compare to desired response. The whole balance response took 121 milliseconds to complete and might cause driver to drop off from the vehicle. So a tuning must be increased in PID controller so that motor is fast enough to response to angle change of vehicle.

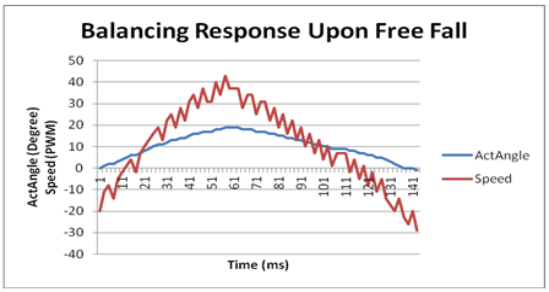

Figure-9. Balancing response plot upon free fall.

First, proportional gain is tuned first. By expecting 30 percent value from set point, the first tuning data was start with P=9, I=0 and D=0. Though, PID tuning was not as easy as seen, multiple tries had been tested on P values where oscillation does happened but does not smooth enough to perform self-balance. After trying with P=4.5, P=2.25 and P=3.375, the last results showed balance response that is consider satisfying. According to manual tuning method, the next tuning parameter is Integral gain which it reduces overshoot of motor response. As manual tuning method does, the tuning of Integral gain starts from I=0.1 with an increment of 0.05 for each test. The final satisfying results is the value of 0.5 where the vehicle is able to balance with less overshoot and more stable. For the last parameter test, the method is same as Integral gain tuning where Derivatives gain works in damping the response of oscillation towards change of angle. The Derivative gain finalize is 2.5 and finally the whole program will need to tuned with overall gain in order to archive complete response of PID. For this part, the final tuned overall gain is 1.5 where the controller acts to perform balance with the vehicle but with some gaggling motions when driver rides. Figure-11 illustrates the plots of balancing response upon free fall after PID tuning. When comparison of Figure-9 and 10 is made, it is clearly shown that the response apparently improves by the response time of motor speed towards angle changes. The response start since there is angle changes and response to balance for one cycle within 81 milliseconds. Compared to Figure-9, response of Figure-10 is more desirable and effective by the sensitivity of response and speed.

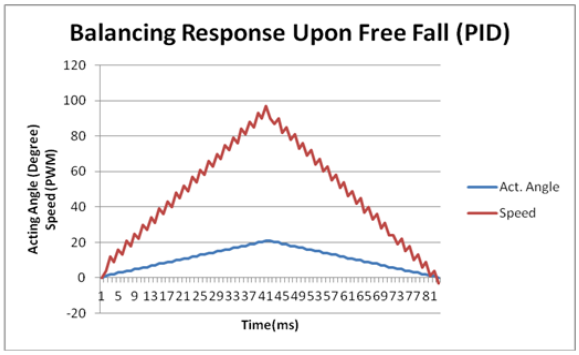

Figure-10. Balancing response plot upon free fall (PID).

After PID was tuned, balancing response was tested under angle limitation where vehicle will stop to response once acting angle is over the set angle. By refer to Figure-11, a plot was done by limiting acting angle of balancing response and speed response was monitored. It is showed that upon increment of angle over 20 degrees, motor will stop react causing stop to vehicle and indicates driver that the tilt angle is not secured.

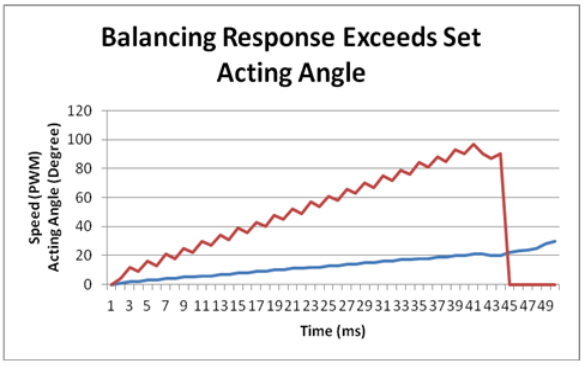

Figure-11. Balancing response plot exceeds set acting angle.

Tuning of PID gains in the program was satisfying but was not the best. It is able to balance driver while riding on it but the shagging of balance process does not comforts drivers to ride on it. After the vehicle was able to balance, comparisons are made with real market product so that prototype is being proved compact. By compare to Segway i2, a commercialized product, the specifications can be seen through Table-1. The prototype specification, are smaller and lighter compare to real products, so the compact design objective was achieved.

Table-1. Prototype of Segway i2.

Specification	Prototype	Segwayi2
FootPrint	50 cm c 50 cm	48 cm x 63cm
Tire Dimension	16 cm	48 cm
Weight	20 kg	47.7 kg
Material	Wood & Metal	Metal & Plastic

CONCLUSIONS

Development of Self Balancing Two Wheel Electric Vehicle using Arduino is a new approach for the personal transporter. Besides from transport human as loads, the vehicle also designed with compact and lighter weight that consumes less space and energy to carry. Movement of the vehicle resolves balancing terms that is static with a moveable balancing vehicle that is able to carry driver with the algorithm of combining Kalman Filter and PID controller. In terms of short moving distances, having a two wheel self-balancing vehicle does supports travel within short distances as it does not consume energy and eco-friendly. Meanwhile, the cost of production is considered low compared to real actual product.

As a conclusion, the development of this project is managed to achieve all objectives. The implementation of this low cost self-balancing vehicle is encouraged where it leads to eco-friendly by the battery used without harming environment and able to own by everyone with a smaller compact sized vehicle.

REFERENCES

[1] Petrov, P. and Parent M. (2010). Dynamic Modeling and Adaptive Motion Control of a Two Wheeled Self Balancing Vehicle for Personal Transport. 13th International IEEE Annual Conference on Intelligent Transportation Systems.

[2] B. He, W.Z. Liu and H.F Lv (2010).The Kinematics Model of a Two-wheeled Self-balancing Autonomous Mobile Robot and Its Simulation. Second International Conference on Computer Engineering and Application.

[3] H.S. Juang and K.Y. Lum (2013).Design and Control of a Two-Wheel Self-Balancing Robot using the Arduino Microcontroller Board. 10th IEEE International Conference on Control and Automation (ICCA).

[4] J. Wu and W. Zhang (2011).Design of Fuzzy Logic Controller for Two-wheeled Self-balancing Robot. The 6th International Forum on Strategic Technology.

[5] S. Miasa, M. Al-Mjali *et al. (*2010).Fuzzy Control of a Two-Wheel Balancing Robot using DSPIC.7th International Multi-Conference on System, Signal and Devices.

[6] U. Nagarajan (2013).Fast and Graceful Balancing Mobile Robots.ProQuest LLC.

[7] P.T. Chen (2010).Simulation and Optimization of a Two-Wheeled, Ball-Flinging Robot", ProQuest LLC.

[8] A. Salerno (2006).Design, Dynamics and Control of a Fast Two-Wheeled Quasiholonomic Robot.Library and Archives Canada.

[9] Starlino (2013).Arduino Code for IMU Guide Algorithm. Retrieved from http://www. starlino. com/ imu_ kalman_arduino.html

[10] Kas (2013). Balancing Robot for Dummies. Retrieved from http://www.x firm.com/?page_id=145

[11] S.Colton (2013).The Balance Filter. Retrieved from http://web.mit. edu/ scolton/ www/ filter.pdf

[12] Alegiaco (2013).Kalman Filter vs Complementary Filter. Retrieved from http:// lets make robots. com/node/29121

[13] Sensor and Sensing (2013). Retrieved from http:// cs. brown. edu/~ tld/ courses/ cs148 / 02/sensors.html

[14] Introduction to AI (2013). Retrieved from http:// www. ru.is/ faculty /thorisson /courses /v2008/gervigreind/FuzzyLogic.html

[15] Kalman Filtering of IMU Data (2013) .Retrieved from http:// tom. pycke. be/ mav/ 71/ kalman-filtering-of-imu-data